View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositorio Digital de la Universidad Politécnica de Cartagena

# PORTABILIDAD DE LOS GRAFICOS INTERACTIVOS PARA EL DISEÑO DE ESTRUCTURAS EN EL SISTEMA DISSENY

**Pedro Company Calleja**
Universidad Jaume I
Dpto. de Tecnología
Campus de Penyeta Roja. E-12071, CASTELLON. SPAIN

**Pascual Martí Montrull**
Universidad de Murcia
Dpto. de Ingeniería Electromecánica
Paseo Alfonso XIII,48. E-30203, CARTAGENA. SPAIN.

**Vincenzo Caiozzo**
Università di Catania (Becario ERASMUS en Univ. Politécnica de Valencia)
Instituto di Informatica e Telecomunicazioni.
V.le A. Doria, 6. I-95125, CATANIA, ITALY.

**Resumen:** Las técnicas CAD están ya firmemente establecidas en los sistemas de Diseño Optimo de estructuras. En particular, las que dotan al diseño optimo de estructuras de toda la capacidad de comunicación de los gráficos, y todos los recursos de interacción con el usuario.
Sin embargo, las librerías gráficas sobre las que estas técnicas CAD se han desarrollado, han experimentado una gran evolución. En consecuencia, los sistemas de Diseño Optimo de estructuras han llegado a sufrir una o mas *migraciones* conforme evolucionaban los estandares gráficos. Además, la aparición de nuevos, y mas potentes, estandares de facto, hace previsible que persista la necesidad de nuevas migraciones. Por ello, el mantenimiento y desarrollo de sistemas que utilizan masivamente las técnicas CAD, requiere una estrategia de maximización de la *independencia* de las aplicaciones respecto a las librerías gráficas.
En la presente comunicación se describe la evolución de la interacción gráfica del sistema de diseño óptimo de estructuras DISSENY [Martí, 1985]. En la descripción se destacan la solución finalmente adoptada de implementar las operaciones gráficas mas complejas (tales como la gestión de menús, la visualización de modelos y la representación de información no geométrica), y utilizar una interfase a una librería virtual, con sintaxis GKS, para las operaciones mas elementales (transformaciones, representación de primitivas y entradas interactivas).

## 1. INTRODUCCION

Las técnicas CAD, entendidas en su sentido mas amplio de *diseño asistido* por ordenador, están firmemente establecidas en los sistemas de Diseño de Estructuras. Dichas técnicas comprenden todos los aspectos que facilitan el uso del ordenador en el diseño. Siendo los aspectos mas destacados la definición e implementación de *Bases de Datos* específicas para diseño de estructuras, y la potenciación de la comunicación interactiva entre el usuario y el sistema de diseño, por medio de la *Interfase Gráfica de Usuario* apropiada.

Las técnicas de gestión de datos, aplicadas a sistemas de diseño, así como su particularización en el caso del sistema Disseny, ya fueron tratadas en una ponencia anterior [Company, 1987]. Por su parte, las técnicas que dotan al sistema de diseño de toda la capacidad de comunicación de los gráficos y todos los recursos de interacción con el usuario, también fueron expuestas [Company,1988] [Company,1990]; si bien la implementación de las mismas en el sistema DISSENY ha sufrido posteriores modificaciones sustanciales en los aspectos de interección y representaciones gráficas.

Las modificaciones y ampliaciones que han marcado la evolución del sistema, han ido emparejadas con diferentes *migraciones*, tanto de las herramientas lógicas (software) como de las máquinas sobre las que el sistema ha estado instalado. Las migraciones de máquina han sido necesarias por dos motivos:

- adaptar el sistema, personalizandolo y acoplandolo a los recursos de los clientes, y
- utilizar máquinas mas capaces y/o veloces, para potenciar las prestaciones del sistema.

Las migraciones de máquina siempre han implicado un cambio de compilador y montador. Aunque dada la restricción voluntaria de programar asumiendo el estandar de Fortran, no ha significado un cambio sustancial (pese a que el propio Fortran ha evolucionado desde el Fortran V al Fortran 77, y ahora está introduciendose el aún no estandar Fortran 90).

Pero las migraciones de máquina también han forzado el cambio de librería gráfica. En este caso, el cambio ha sido siempre mas drástico debido, fundamentalmente, a la no disponibilidad de la misma librería gráfica en las diferentes máquinas utilizadas. Así, el módulo de análisis original disponía de un rudimentario postprocesador gráfico montado sobre la librería gráfica PLOT10. La migración del sistema para poder ejecutar el módulo de análisis en ordenadores personales de primera generación (8086 y 80286) requirió la necesaria adopción de una librería gráfica diferente, al no estar PLOT10 disponible en las nuevas plataformas. La librería elegida fue HALO, dado que era una librería accesible y de bajo costo, y estaba disponible para todo tipo de ordenadores personales y una aceptable variedad de periféricos.

La falta de adaptación de los ordenadores "mainframe" para la interacción gráfica, se hizo mas patente con la popularización de los PC's, y sus crecientes prestaciones gráficas, que, no obstante, contrastaban con su escasa capacidad de cálculo. Esta situación propició que la siguiente generación del sistema DISSENY tubiese módulos de cálculo que se ejecutaban en un gran ordenador de cálculo y módulos de visualización que se ejecutaban en ordenadores personales (con la interconexión garantizada por la modularidad del sistema y la base de datos adoptada). Los módulos de representación mencionados seguian estando montados sobre la librería HALO.

La aparición de las estaciones de trabajo, que aunaban una gran capacidad de cálculo con unas crecientes prestaciones gráficas fue la causa de la siguiente migración. En esta nueva ocasión, ya existía una librería gráfica estandar: Graphical Kernel System (GKS) [Hopgood,1986]. Pese a que las implementaciones disponibles para dicha librería presentaban "defectos" bastante importantes (p.e. la incompleta y lenta gestión de segmentos), el hecho de disponer de una misma librería para diferentes plataformas [HP-GKS,1989] [GSS*GKS,1989], decidió la elección.

El último "salto" ha sido desde las mencionadas librerías GKS comerciales hasta una librería gráfica propia con sintaxis GKS y montada sobre la librería de bajo nivel disponible con el compilador utilizado. Esta migración se ha debido al deseo de "independizar" el sistema de diseño, y en particular todos los módulos y herramientas gráficas que se han ido desarrollando a su alrededor (con un total estimado de 12.000 instrucciones Fortran).

## 2. GRAFICOS EN EL PROCESO DE DISEÑO

Las necesidades de representaciones gráficas en un sistema de diseño son muy distintas en cada una de las tres tareas en las que predomina su uso:

. preproceso,
. monitorización de la evolución, y
. postproceso.

Todas ellas requieren diferentes recursos tanto de *interacción gráfica*, como de *representaciones gráficas*. La distinción, entre interacción y representación gráfica, pretende remarcar la total subordinación de ciertos gráficos al objetivo inmediato de facilitar un diálogo hombre-máquina (p.e. un menú de iconos), frente a la utilización de una representación gráfica "solo" para informar al usuario de algún aspecto del proceso (p.e. la monitorización de la evolución de la función objetivo). En éste último caso la representación no tiene como objetivo inmediato y fundamental provocar una acción por parte del usuario. Aunque, en general, son éste tipo de representaciones las que permiten que el usuario tome decisiones, y, en consecuencia, solicite acciones al sistema (a través de la Interfase Gráfica de Usuario).

Por lo que respecta a las representaciones gráficas empleadas en cada una de las tareas de diseño en las que intervienen, se debe comenzar remarcando que los preprocesadores para análisis fueron los precursores de los sistemas CAD, entendidos estos como sistemas de DIBUJO asistido por ordenador. Pero las dependencias geométricas habituales en los modelos de diseño, hacen que los preprocesadores para DISEÑO no sean meras extensiones o actualizaciones de los bien establecidos preprocesadores para análisis. Esto es debido, por un parte, a que el modelo geométrico ya no puede corresponder a una geometría fija (pues, obviamente, en tal caso el algoritmo de optimización carecería de variables para buscar la mejor geometría). Por otra parte, la definición de una geometría dependiente de variables pasa por la elección, no trivial de la estrategia de definición del modelo de análisis a partir del de diseño, y las posteriores actualizaciones de ambos sin perder la coherencia.

La monitorización es un tema que ha sido introducido recientemente en el campo del diseño óptimo de estructuras [Fleury,1986] [Company,1992], y no existen criterios firmemente establecidos sobre que información debe monitorizarse. Si bien se apunta la distinción entre representación de la *evolución* del proceso (función objetivo, variables y restricciones básicamente) y la representación de diferentes *subespacios de diseño.* En cualquier caso, se trata de representar información no geométrica, y se requiere además la facilidad de controlar interactivamente tanto la información que se representa como el modo en que se hace.

Los postprocesadores gráficos para el análisis de estructuras están perfectamente establecidos (p.e. [Quiroz,1989]). Pero, en este caso, si que cabe destacar que estos postprocesadores son básicamente los mismos que requiere un postproceso de diseño.

## 3. LIBRERIA GRAFICA DEL SISTEMA DISSENY

Los gráficos empleados en los procesos de diseño descritos son muy variados. Existen paquetes comerciales y/o librerías gráficas bien adaptadas a:

- modelado geométrico,
- visualización de información (monitorización), y
- creación de entornos interactivos (gestores de menus y ventanas).

Pero un sistema que use todas esas libreria será necesariamente voluminoso, y, lo que es mas importante, será "dependiente de software". La alternativa es crear una librería gráfica propia, que dote al sistema de las necesarias herramientas gráficas (que no son sino un subconjunto de las disponibles en los paquetes comerciales). Esta opción es costosa, en cuanto que requiere desarrollar software gráfico, pero este costo queda compensado porque el cliente no queda obligado a adquirir paquetes de programas complementarios al propio sistema.

Otra ventaja fundamental de desarrollar herramientas gráficas propias es la posibilidad de conseguir un *estilo de interacción* único para todo el sistema de diseño.

Por otra parte, si las librerías propias son de alto nivel (librerías capaces de realizar tareas bastante complejas a partir de pocas instrucciones; fig. 1), y se construyen sobre una librería de bajo nivel ajena, el costo del desarrollo se reduce mucho, sin encarecer apenas el producto final. Veremos que una estrategia de selección correcta de la librería de bajo nivel, y la definición de la interfase apropiada entre la misma y la librería de alto nivel, pueden conseguir una solución eficiente y portable.

La librería gráfica del sistema DISSENY, desarrollada con software propio, y basada en librerías gráficas de bajo nivel, consta de los módulos siguientes:

- Visualización de modelos tridimensionales, con selección interactiva de todos los parámetros de visualización [Company,1988].
- Representación de funciones [Company, 1990].
- Gestión de menús (según una jerarquía de menús desplegables) e interacción (cajas de diálogo).

El primer módulo se programó cuando el sistema utilizaba una librería 2D (PLOT10), haciendo innecesaria la posterior adopción de librería con capacidad 3D (p.e. PHIGS). Además debe desacarse

que el módulo esta conectado al sistema ha traves de una interfase controlada interactivamente por el usuario, para facilitar la representación de la información de diseño asociada al modelo geométrico (deformadas, mapas de esfuerzos y tensiones, etc).

El módulo de representación de funciones se desarrolló dada la imposibilidad de disponer de los fuentes de las librería comerciales existentes en aquel momento (p.e. IMSL). La accesibilidad de los códigos fuente era un requisito de partida para poder construir la interfase interactiva que permite controlar la visualización de la información seleccionada por el usuario, de forma óptima y con los atributos apropiados (tipo de gráfica, códigos de color, etc). En la figura 1 se ve un ejemplo de la utilización del módulo desde el resto de la aplicación. Nótese que la representación propiamente dicha de una tabla, se consigue con una sola operación (FUNCIO), siendo el resto de operaciones meras inicializaciones de todos los artibutos (que solo tienen que redefinirse cuando se modifican). Nótese también la sintaxis estandarizada de todas las operaciones de asignación de atributos.

```
c     Inicializacion
c     ==============
c     Atributos de los ejes
      CALL dats_fun ('EJES',nul_e ,nul_r,'SI' ,cero,cero,1)
      CALL dats_fun ('TTEJ',tipoT ,nul_r,nul_a,1,cero,cero) ! Tipo de texto
      CALL dats_fun ('CTEJ',colorT,nul_r,nul_a,1,cero,cero) ! Color de texto
      CALL dats_fun ('TLEJ',tipoL ,nul_r,nul_a,1,cero,cero) ! Tipo de linea
      CALL dats_fun ('CLEJ',colorL,nul_r,nul_a,1,cero,cero) ! Color de linea

c     Atributos de la funcion
      CALL dats_fun ('FUNC',nul_e ,nul_r,'SI' ,cero,cero,1)
      CALL dats_fun ('TPFU',tipoP ,nul_r,nul_a,1,cero,cero) ! Tipo de puntos
      CALL dats_fun ('CPFU',colorP,nul_r,nul_a,1,cero,cero) ! Color de puntos
      CALL dats_fun ('TLFU',tipoF ,nul_r,nul_a,1,cero,cero) ! Tipo de lineas
      CALL dats_fun ('CLFU',colorF,nul_r,nul_a,1,cero,cero) ! Color de lineas

c     Atributos de las etiquetas (valor numerico) de los puntos
      CALL dats_fun ('ETIQ',nul_e,nul_r,'NO',cero,cero,1)

c     Rangos para graduacion de los ejes
      rangox(1)=  minimoX
      rangox(2)=  maximoX
      CALL dats_fun ('RANX',nul_e,rangox,nul_a,cero,2,cero)
      rangoy(1)=  minimoY
      rangoy(2)=  maximoY
      CALL dats_fun ('RANY',nul_e,rangoy,nul_a,cero,2,cero)

c     Rotulos de la representacion
      CALL dats_fun ('ROTU',nul_e,nul_r,'Funcion Objetivo',cero,cero,1)
      CALL dats_fun ('EJEX',nul_e,nul_r,'Iteraciones'     ,cero,cero,1)
      CALL dats_fun ('EJEY',nul_e,nul_r,'Funcion'         ,cero,cero,1)

c     Representacion de la funcion
c     ============================
      CALL funcio (tabla,      ! Tabla con los valores de las funciones
     +             tipo,       ! Tipo de tabla
     +             nfun,lifun, ! numero y lista de funciones de la tabla
                               ! que se van a representar
     +             nvar,livar  ! numero y lista de valores de la variable
                               ! para los que se representa la funcion
     +             ventana,    ! Identificacion de la ventana
     +             factor,     ! Factor de escala de la representacion
     +             borra,ierr) ! Bandera de borrado y bandera de error
```

**Figura 1**. Ejemplo de utilización de la librería gráfica del sistema DISSENY: Representación de la evolución de la función objetivo.

Por último, el desarrollo del gestor de menús e interacción se anticipó a la disponibilidad de compiladores con extensiones "visual". Es decir, compiladores de lenguajes ampliados respecto a los

correspondientes estandares para incluir funciones que facilitan enormemente la construcción de Interfaces Gráficas de Usuario (GUI's).

La utilidad de todos estos recursos queda patente en la figura 2, en donde se muestra una imagen de la pantalla obtenida durante una sesión de diseño óptimo de una ejemplo típico de optimización de estructuras. El modelo de diseño del voladizo aparece representado en la ventana grande de la derecha; con él está asociado el mapa de colores de la pequeña ventana inferior derecha (las barras del modelo se representan con sus características de diseño codificadas por colores, que en la figura no pueden observarse). En cuatro de las ventanas de la mitad izquierda se representan las evoluciones de diferentes valores asociados al proceso de diseño (función objetivo, variables y restricciones). En las dos ventanas inferiores se muestran las sensibilidades de la función objetivo y de una de las restricciones mas violadas, respecto a cada variable. Por último, rodeando a las anteriores ventanas, se situan las barras de menús (arriba), que dan acceso a los menús desplegables, y el menú principal de utilidades (derecha) y su submenú activo.

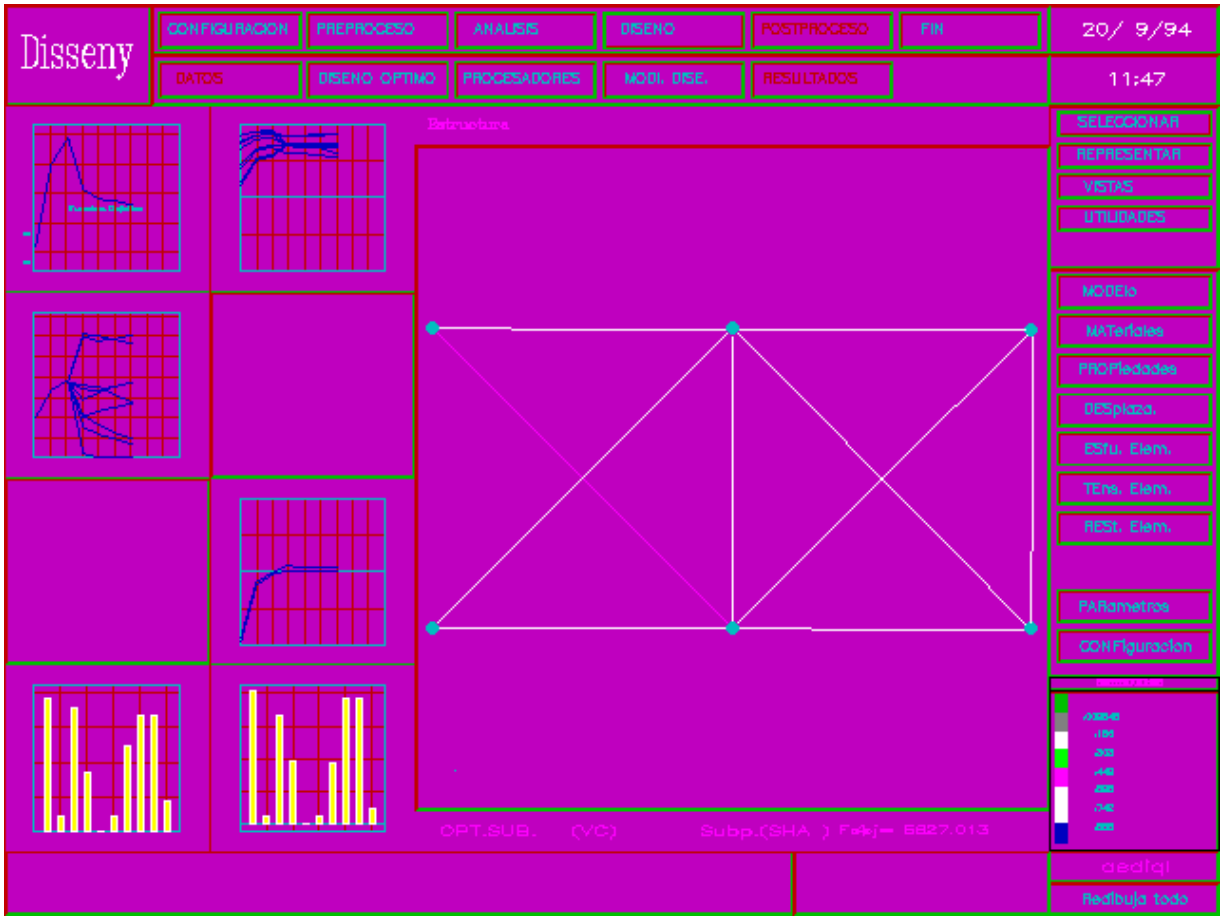


**Figura 2.** Imagen de la pantalla durante una sesion de diseño óptimo en el sistema DISSENY

## 4. LIBRERIA GRAFICA DE BASE DEL SISTEMA DISSENY

Es importante destacar que las sucesivas migraciones que ha sufrido el sistema DISSENY han forzado al mismo a utilizar un "mínimo común" de las sucesivas librerías sobre las que se ha instalado. Este mínimo común puede resumirse en:

- Operaciones de salida gráfica:
  **.** Representación de primitivas (marcador, segmento, texto y relleno de áreas poligonales).
  **.** Gestión de transformación window-viewport, para la representación de una porción del mundo real (2D) sobre una región del periférico de salida.
- Operaciones de entrada con realimentación gráfica: gestión de entradas por teclado y/o ratón.

El mínimo común de operaciones descrito arriba es el que se estaba utilizando en la versión precedente del sistema, que estaba montada sobre GKS. Así, aunque la librería GKS permite manejar diferentes estaciones de trabajo simultaneamente, y permite manejar conjuntos de primitivas (segmentos), estas operaciones de mayor nivel no estaban siendo utilizadas. Es decir, se estaba utilizando un subconjunto de la librería gráfica GKS.

La ventaja de utilizar GKS consiste en disponer de un conjunto de operaciones gráficas bien definidas por medio de una sintaxis estandar, que facilitan el trabajo del programador de la librería de alto nivel. El inconveniente es la necesidad de disponer de una implementación de una librería GKS apropiada tanto para la máquina como para el compilador utilizados.

## 5. EMULACION DE UN SUBCONJUNTO DEL ESTANDAR GKS

Tal como se ha explicado arriba, para mantener la independencia de la librería gráfica de alto nivel, como para facilitar la portabilidad del sistema, se decidió:

- mantener la sintaxis GKS en la utilización de la librería de bajo nivel, por parte de la librería de alto nivel, y
- emular la librería de bajo nivel, por medio de una interfase desde la sintaxis GKS a las operaciones de la librería de bajo nivel empleada realmente (GRAPHICS de Fortran PowerStation, [Fortran,1993]).

Manteniendo la sintaxis GKS [Sproull,1985] se evita la migración de la librería de alto nivel, al tiempo que se mantiene una alta legibilidad y robustez de la misma. Mientras que al emular dicha sintaxis, disponemos de modulos de emulación intercambiables que aseguran una rápida portabilidad del sistema.

Para crear la interfase de emulación GKS, decidimos comenzar por un emulador "identidad", que recibía las llamadas GKS de la librería de alto nivel (en donde se modificaron los nombres de las operaciones GKS por la simple adición del prefijo "u_"; de forma que GQDSP se conviertió en U_GQDSP) y las convertía en llamadas a las mismas operaciones de la librería GKS (así U_GQDSP se convierte en GQDSP, y mantiene los mismos argumentos). El propósito inicial de este emulador identidad era simplemente mantener la posibilidad de utilizar las versiones precedentes. Pero debe destacarse que disponer de una interfase permite ampliar las posibilidades de la librería GKS, haciendo extensiones que quedan *encapsuladas* en la interfase; manteniendo así la independencia de la librería de alto nivel. Tal es el caso del periférico lógico trackball que hemos emulado a traves del periférico físico mouse.

El segundo paso consistió en desarrollar un segundo emulador, que transformaba las llamadas GKS modificadas (u_operacion) en las necesarias operaciones de la librería GRAPHICS que permiten emular la operación pedida. Por supuesto, en la mayoría de los casos, esta conversión no es inmediata, sino que requiere llamar a varias operaciones de la librería GRAPHICS [Fortran,1993] para emular una sola operación GKS (Ver figura 3). Pero siguiendo estrategias de implementación de operaciones gráficas como las descritas en [Harrington,1983], se llega a una interfase no muy voluminosa (unas 2500 instrucciones Fortran para emular 60 operaciones GKS con GRAPHICS) e igual de rápida que la librería GKS real.

```
      SUBROUTINE u_gqdsp (nwork,ierr,iduni,rx,ry,lx,ly)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*         Indica el tamano del display.                                   *
*            nwork   Identificador de la estacion de trabajo (solo    *
*                    se acepta una estación)                            *
*            iduni   indica las unidades device:                        *
*                    0 para metros                                      *
*                    1 para otras unidades                              *
*            rx,ry   son las coordenadas maximas en unidades device.  *
*            lx,ly   son las coordenadas maximas en unidades raster.  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
c     Argumentos
c     ----------
      INTEGER          nwork,ierr,iduni,lx,ly
      REAL*4           rx,ry

c     Variables locales
c     -----------------
      STRUCTURE/videoconfig/       ! Estructura de datos de GRAPHICS
        INTEGER*2 numxpixels       ! number of pixels on X axis
        INTEGER*2 numypixels       ! number of pixels on Y axis
        INTEGER*2 numtextcols      ! number of text columns available
        INTEGER*2 numtextrows      ! number of text rows available
        INTEGER*2 numcolors        ! number of actual colors
        INTEGER*2 bitsperpixel     ! number of bits per pixel
        INTEGER*2 numvideopages    ! number of available video pages
        INTEGER*2 mode             ! current video mode
        INTEGER*2 adapter          ! active display adapter
        INTEGER*2 monitor          ! active display monitor
        INTEGER*2 memory           ! adapter video memory in K bytes
      END STRUCTURE
      RECORD /videoconfig/ screen

c     Inquisicion de las dimensiones de la pantalla
c     *********************************************
      CALL getvideoconfig (screen) ! Llamada a la libreria GRAPHICS
      lx= INT(screen.numxpixels)
      ly= INT(screen.numypixels)

      rx   = FLOAT(lx)
      ry   = FLOAT(ly)
      iduni= 1            ! las coordenadas device tambien son pixels

      ierr = 0

c     Finalizacion
c     ************
      RETURN
      END
```

**Figura 3**. Emulación de la operación de inquisición del tamaño del periférico pantalla.

## 6. CONCLUSIONES

Se han discutido las posibilidades del empleo de librerías gráficas para implementar las representaciones gráficas y las Interfaces Gráficas de Usuario que precisa un sistema interactivo de diseño óptimo de estructuras. Se ha destacado la variedad de tareas que implica un proceso de diseño: preproceso, monitorización del diseño y postproceso. Esta diversidad hace que la alternativa de utilizar librerías comerciales de lugar a implementaciones caras, dependientes de software y voluminosas. En el extremo opuesto, el desarrollo de librerías propias es muy costoso.

La solución intermedia es desarrollar una librería propia de alto nivel (para gestión de menús, visualización de modelos 3D y representación de funciones) basada en una librería de bajo nivel ajena. Esta solución tiene un costo de desarrollo razonable, dota al sistema de interfaces para una

aceptable variedad de periféricos, y simplifica las migraciones de máquina y/o compiladores u otras herramientas lógicas. Si, además la librería de bajo nivel se conecta con la librería de alto nivel por una interfase que emule la sintaxis GKS, se obtiene un código legible y robusto que maniene encapsuladas todas las dependencias de software, simplificando mucho las adaptaciones necesarias para nuevas migraciones.

## 7. REFERENCIAS


Company P. y Martí P. (1987)
Organización y manejo de información en el sistema de diseño óptimo de estrtucturas DISSENY.
*Anales de Ingeniería Mecánica*. Año 5, nºII, 1987. p 33-38.

Company P. y Martí P. (1988)
Aplicación de las técnicas CAD para el análisis y diseño interactivo de estructuras en el sistema DISSENY.
*Anales de Ingeniería Mecánica*. Año 6, nºIII, 1988. p 275-280.

Company P. y Martí P. (1990)
Representación gráfica del proceso de diseño óptimo de estructuras en el sistema DISSENY.
*Memorias del I congreso de Métodos Numéricos en Ingeniería*.
Ed. G.Winter y M.Galante, Barcelona, 1990, p 779-786.

Fleury C. (1986)
Computer Aided Optimal Design of Elastic Structures.
*Computer Aided Optimal Design*. Ed. Springer-Verlag. 1986. p 831-900

*GSS*GKS Kernel System. Programmer's Guide*.
Graphic Software Systems, Inc. Versión 2.12 (1989).

Harrington S. (1983).
*Computer Graphics. A Programming Approach*.
Ed. McGraw-Hill. 1983

Hopgood F.R.A.; Duce D.A.; Gallop J.R. y Sutcliffe D.C. (1986).
*Introduction to the Graphical Kernel System (GKS).*
Ed. Academic Press. 1986.

*HP-GKS User's Guide. HP 900 Series 300/800 Computers*.
Hewlett-Packard Company, (1989)

Martí P. y Company P. (1985)
DISSENY. Un sistema interacivo para el diseño de estructuras basado en técnicas de optimización.
*Anales de Ingeniería Mecánica*. Año 3, nº I, 1985. p 285-290.

*Microsoft Fortran PowerStation. Language Guide*.
Microsoft Corporation, (1993)

Quiroz L. y Dufeu E. (1989)
Un postprocesador gráfico independiente de los dispositivos
*Revista Internacional de Métodos Numéricos par Cálculo y Diseño en Ingeniería*. Vol. 5, 2, 1989. p263-275.

Sproull R.F.; Sutherland W.R. y Ullner M.K. (1985).
*Device-Independent Graphics (With examples from IBM personal computers).*
Ed. McGraw-Hill. 1985.